OPENSCIS

Journal of Experimental and Computational Methods in Applied Sciences, pp. 1–15

Multi-Agent Reinforcement Learning with Swarm Coordination for Dynamic Resource Provisioning in Enterprise Analytics Platforms

Marcos Vinícius Almeida [1], Isabela Rocha Monteiro [2]

1. *Department of Computer Science, Southern Coast Federal University, 855 Avenida das Gaivotas, Ponta da Praia, Santos 11030-500, Brazil*
2. *Department of Information Systems, Federal Institute of Technology of Paraná, 422 Rua Engenheiro Rebouças, Rebouças, Curitiba 80215-900, Brazil*

Abstract

Enterprise analytics platforms expose shared pools of compute, memory, and storage resources to heterogeneous workloads such as interactive queries, streaming pipelines, and batch jobs. These workloads exhibit strong non-stationarity due to diurnal patterns, business events, and evolving user behavior, which makes static or rule-based resource management brittle and often inefficient. At the same time, enterprises impose service level agreements on latency, throughput, and availability that need to be satisfied under cost and energy constraints. Conventional autoscaling controllers rely on local metrics and hand-tuned thresholds, and they often treat the platform as a monolithic system, ignoring the spatial structure of clusters and the coupling between tenants. Multi-agent reinforcement learning provides a way to learn adaptive policies from interaction, while swarm coordination offers a decentralized mechanism for aligning behavior across large agent populations. This paper studies dynamic resource provisioning in enterprise analytics platforms through a swarm-coordinated multi-agent reinforcement learning formulation. The platform is modeled as a set of interacting resource pools, each controlled by an agent that observes local performance signals and takes scaling actions. Swarm coordination mechanisms propagate aggregate load and congestion information across agents using lightweight neighborhood communication, which reduces oscillations and improves global constraint satisfaction without centralizing the control logic. A linear model of resource consumption and performance is integrated into the learning process to structure value functions and constrain exploration. The study investigates stability, scalability, and empirical behavior of the proposed framework across a range of workload patterns, highlighting trade-offs between responsiveness, resource utilization, and service level adherence.

Introduction

Enterprise analytics platforms combine data warehousing, stream processing, and large-scale batch computation into a single environment that serves many teams and business units [1]. Typical deployments rely on container orchestration systems or virtualized clusters, where nodes host multiple services and query engines. Workloads in such platforms range from latency-sensitive dashboard queries to long-running model training jobs, and their arrival processes can exhibit strong burstiness driven by human activities, external events, and automated data pipelines. These characteristics create a persistent tension between overprovisioning resources to protect performance and underprovisioning to reduce infrastructure cost and energy consumption [2].

Dynamic resource provisioning aims to adjust allocations of CPU, memory, and I/O capacity in response to observed load, while satisfying service level objectives on latency, throughput, and error rates. Classical control techniques and heuristic autoscaling policies use metric thresholds, moving averages, or simple predictive models to decide how many replicas of a service should run in each time window. While such mechanisms are relatively simple to deploy, they depend heavily on manual tuning, and their parameters often need continual adjustment as workloads and hardware evolve [3]. Moreover, many heuristics treat each service or pool in isolation, which can lead to resource contention, synchronized oscillations, and inefficient use of shared capacity across the cluster.

Reinforcement learning offers a way to adapt resource management policies by optimizing long-run cost functions derived from operational metrics such as response time, queue length, and infrastructure expenditure. In a

reinforcement learning formulation, a controller observes the state of the platform, selects provisioning actions, and receives feedback through a reward signal that reflects both performance and cost. However, enterprise analytics platforms are distributed systems with many control points and partial observability [4]. A purely centralized controller faces scalability and robustness challenges, while independent learning agents risk converging to conflicting behaviors that degrade global performance and violate constraints.

Multi-agent reinforcement learning extends single-agent formulations to settings where multiple decision makers interact in a shared environment. Each agent observes a subset of the global state and takes actions that influence both its own performance and that of others [5]. This creates a stochastic game in which agents adapt their policies based on local signals and possibly limited communication. For large clusters, a multi-agent perspective is natural: each resource pool or service group can be controlled by an agent that reasons about its local workload and neighbors. Yet, without additional coordination mechanisms, multi-agent learning can suffer from non-stationary interactions, slow convergence, and unstable policies.

Swarm coordination provides a complementary paradigm in which simple local rules and exchange of low-dimensional signals produce coherent global behavior across many agents [6]. Originally studied in the context of biological swarms, ant colonies, and bird flocks, swarm-inspired algorithms use ideas such as virtual pheromone fields, attraction-repulsion dynamics, and consensus protocols on sparse graphs. These mechanisms can be embedded into multi-agent reinforcement learning to regularize policies, propagate congestion information, and align agents toward globally consistent resource configurations, while preserving decentralized control and communication.

This paper formulates dynamic resource provisioning in enterprise analytics platforms as a swarm-coordinated multi-agent reinforcement learning problem with an explicit linear model of resource and performance dynamics [7]. The platform is represented as a set of nodes or pools, each managed by an agent that selects scaling actions based on local observations and swarm signals aggregated from neighbors. A linear approximation of how allocations influence latency and queue lengths is integrated into the value function representation and reward design, which shapes exploration and supports stability analysis. The study examines how swarm coordination interacts with multi-agent learning across different workload regimes, focusing on metrics such as service level violation rates, aggregate resource usage, and convergence behavior of the learned policies [8].

The remainder of the paper is organized as follows [9]. The next section describes the system model for enterprise analytics platforms, including the workload representation, resource abstraction, and Markov game formulation. A subsequent section introduces the swarm-coordinated multi-agent reinforcement learning framework and its value function structures. The following section develops a linear resource provisioning model and analyzes its integration with the learning process [10]. The experimental section evaluates the behavior of the framework under synthetic and trace-driven workloads. The paper concludes with a discussion of limitations and potential extensions.

System Model for Enterprise Analytics Platforms

The enterprise analytics platform is modeled as a cluster composed of a finite set of nodes, each hosting containers or virtual machines that run analytical engines, query coordinators, stream processors, and auxiliary services. The cluster is partitioned into logical resource pools corresponding to tenants, application domains, or service tiers [11]. Each pool aggregates a subset of nodes and exposes a capacity vector that describes the total compute, memory, and I/O bandwidth available to workloads assigned to that pool. Within a pool, containers implementing specific services are scheduled and scaled up or down over time in response to observed load, but the underlying capacity of the pool is treated as fixed over the time horizons considered here.

Workloads are represented as flows of queries or jobs that arrive to each pool according to stochastic processes [12]. For modeling purposes, arrivals are organized into traffic classes based on latency sensitivity, resource intensity, and priority. Within each class, jobs are processed by one or more services modeled as queues. For an analytics query engine, a query may traverse a front-end planner, an execution engine, and possibly a storage layer. For a stream processing pipeline, records pass through operators arranged in a directed acyclic graph [13]. The model aggregates these detailed paths into effective demand rates and service times per pool, which depend on the resources allocated to the corresponding services.

The state of the platform at discrete decision epochs is described by a vector that aggregates the relevant performance and utilization indicators. Let the dimension of this state space be denoted by a positive integer interpreted as the number of state components [14]. Typical state components include queue lengths or smoothed arrival rates per traffic class, observed latencies over a recent window, and resource utilization fractions per pool. Since full observability of micro-level metrics is not required for control, the state representation is constructed as a low-dimensional summary sufficient for capturing the main dynamics relevant to provisioning decisions.

To express the state more compactly, a vectorial notation is adopted. At decision epoch indexed by a nonnegative integer, the cluster-level state is denoted as [15]

$$x_t \in \mathbb{R}^n$$

where each component corresponds to a particular metric, such as average CPU utilization in a given pool or the

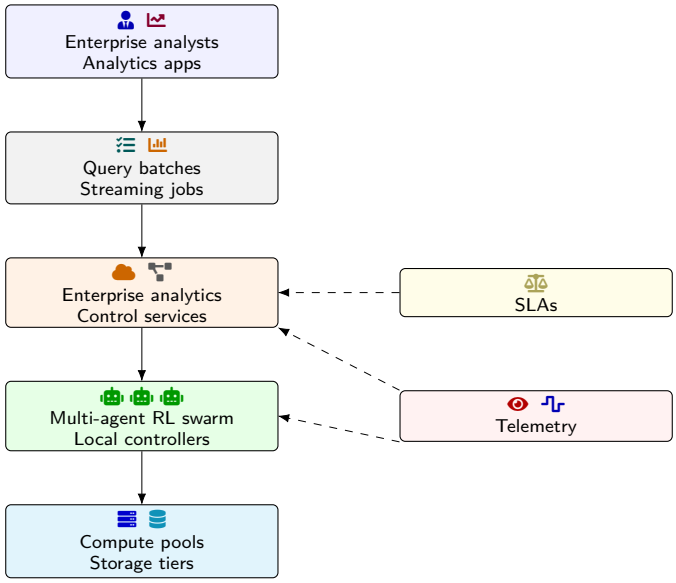

Figure 1: High-level view of a multi-agent reinforcement learning swarm controlling dynamic resource provisioning in an enterprise analytics platform, with user workloads translated into control actions under SLA and telemetry feedback.

Table 1: Workload profiles evaluated in the enterprise analytics platform

Workload type	Query mix	Arrival pattern	Latency SLA (ms)
OLTP-style	90% point, 10% range queries	Poisson, low burst	50
Batch ETL	Large scans, joins	Periodic, every 30m	5000
Streaming analytics	Sliding-window aggregations	High burst, diurnal	200
Ad-hoc analytics	Complex joins, subqueries	Heavy-tail inter-arrival	1000

backlog of a specific latency class. The resource allocation at time index is represented by a vector

$$u_t \in \mathbb{R}^m$$

whose components represent scaling actions, for example the change in the number of replicas of a service or the adjustment of concurrency limits [16]. The feasible set of allocations is constrained by the capacity of each pool and global platform limits. This set is denoted

$$\mathcal{U} = \{u \in \mathbb{R}^m : A_u u \leq b_u\} \tag{1}$$

where the matrix and vector encode capacity and policy constraints. The inequalities are interpreted componentwise [17].

The workload dynamics and their dependence on resources are complex and nonlinear. For analytical tractability and to support linear reinforcement learning structures, a local linear approximation of the state evolution under scaling actions is used. This approximation is constructed around typical operating points of the platform and captures how queue lengths and utilization respond to small increments or decrements in capacity [18]. The state evolution is written as

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{2}$$

where the matrix represents the linearized endogenous dynamics of the state, the matrix describes how actions influence the state, and the disturbance vector represents exogenous fluctuations due to random arrivals and service time variability. The disturbance is treated as a zero-mean random vector with bounded second moments, reflecting variability but not systematic drift in the approximation.

The service level performance of the platform is summarized by a vector of metrics derived from the state [19]. For instance, smoothed tail latencies, throughput per class, and backlog thresholds can be represented as linear or affine functions of the state. Let this performance vector be written as

$$y_t = Cx_t + d \tag{3}$$

where the matrix selects and combines state components, and the vector encodes constant offsets such as baseline latencies that do not depend on current congestion [20]. Service level objectives typically prescribe upper bounds on certain components of this performance vector, such as maximum tolerable latency for interactive queries. These constraints can be approximated as linear inequalities of the form

$$Dy_t \leq e \tag{4}$$

which, after substitution, becomes

$$DCx_t[21] \leq e - Dd \tag{5}$$

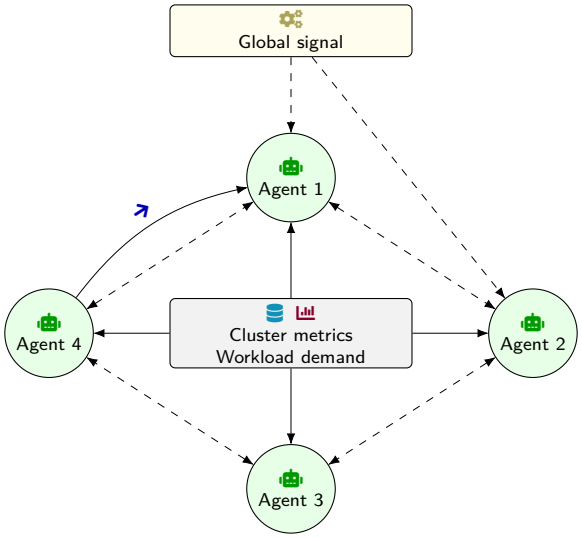

Figure 2: Swarm coordination pattern where each reinforcement learning agent observes a shared environment, exchanges lightweight messages with neighbors, and receives an optional global coordination signal.

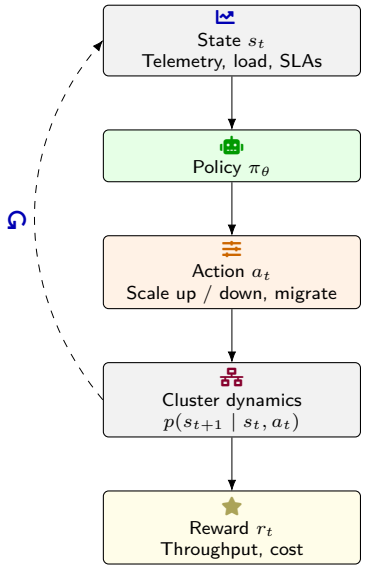

Figure 3: Markov decision process abstraction of dynamic resource provisioning, where the multi-agent policies map platform state into scaling actions that drive cluster dynamics and generate rewards.

so that feasible regions of the state space associated with acceptable performance are described by polyhedral sets.

The cost incurred at time index combines resource usage and penalties for violating service level constraints. A linear stage cost is adopted to maintain compatibility with linear reinforcement learning formulations [22]. Let the cost be given by

$$c_t = p^\top u_t + q^\top \max(0, DCx_t - e + Dd) \tag{6}$$

where the cost vector captures per-unit infrastructure cost of actions, and the vector captures penalty coefficients for each constraint component. The max operator is applied elementwise to represent violations, which are zero when constraints are satisfied and positive otherwise [23]. For the analysis in later sections, a piecewise linear approximation can be employed to maintain the overall linear structure while handling penalties.

Dynamic resource provisioning is formulated as a Markov game in which multiple agents select components of the action vector at each decision epoch. The global state evolves according to the linear dynamics and disturbance process [24]. Each agent corresponds to a particular resource pool or service group and observes a local projection of the global state. Let agent index range over a finite set. Agent observes

$$o_t^i = H_i x_t[25] \tag{7}$$

where the observation matrix selects the components of interest to that agent, such as utilization and queue ratios for services hosted in its pool. Agent chooses a local action

$$u_t^i \in \mathcal{U}_i$$

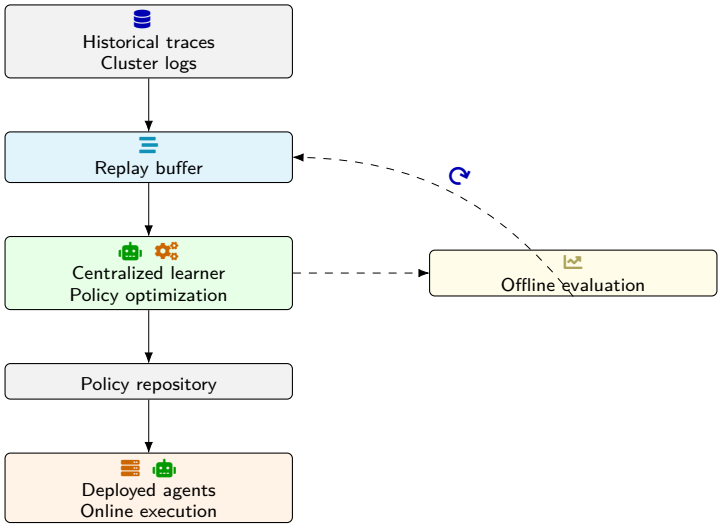

Figure 4: Centralized training pipeline where offline traces feed a replay buffer, a shared learner updates policies, and evaluated models are deployed back into the enterprise analytics platform.

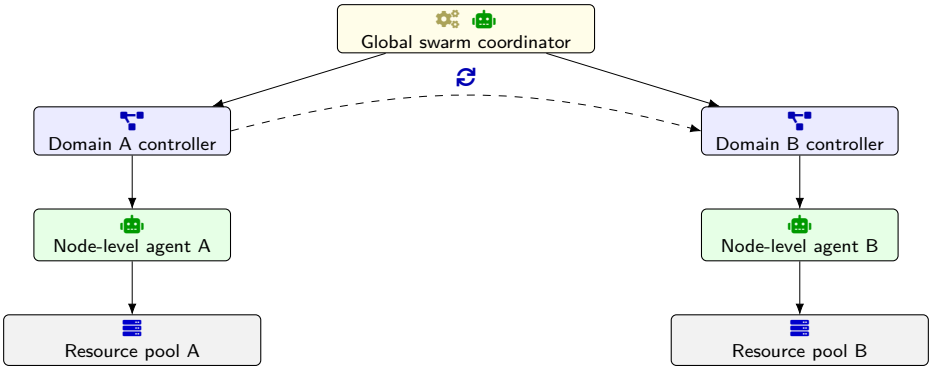

Figure 5: Hierarchical swarm organization with a global coordinator, intermediate domain controllers, and node-level agents that directly actuate resource pools.

and the joint action is the concatenation of all local actions, subject to the global feasibility constraints encoded in the set [26]. The individual contribution of agent to the stage cost can be written as

$$c_t^i = p_i^\top u_t^i + q_i^\top z_t^i \tag{8}$$

where the vectors and define local resource cost and penalty coefficients, and the vector extracts relevant violation signals associated with the services controlled by agent. The total cost is the sum over agents, capturing the aggregate infrastructure expenditure and performance deviations across the platform.

Swarm-Coordinated Multi-Agent Reinforcement Learning

The Markov game described above is approached through a multi-agent reinforcement learning framework in which each agent learns a policy that maps its local observation and possibly a low-dimensional swarm signal to a distribution over local actions [27]. Time is discretized into decision steps, and at each step every agent collects its observation, computes a swarm-coordinated feature vector, selects an action, and then receives a local cost signal. The global state transitions according to the linear dynamics, influenced by the joint action and exogenous disturbance. Learning proceeds over many episodes or long-running interaction traces until policies stabilize [28].

To represent policies and value functions, each agent uses function approximation over a shared linear feature space. Let the local state-action feature mapping for agent be denoted by

$$\phi^i(o_t^i, [29] u_t^i) \in \mathbb{R}^k \tag{9}$$

where denotes the feature dimension. The feature vector may include normalized utilizations, queue ratios, smoothed historical actions, and swarm signals aggregated from neighboring agents. A linear action-value function for agent is defined as [30]

$$Q_{\theta^i}^i(o_t^i, u_t^i) = \phi^i(o_t^i, u_t^i)^\top \theta^i [31] \tag{10}$$

where the parameter vector contains weights learned from experience. The use of a shared feature mapping across agents facilitates transfer of knowledge and supports consistent behavior patterns across the cluster.

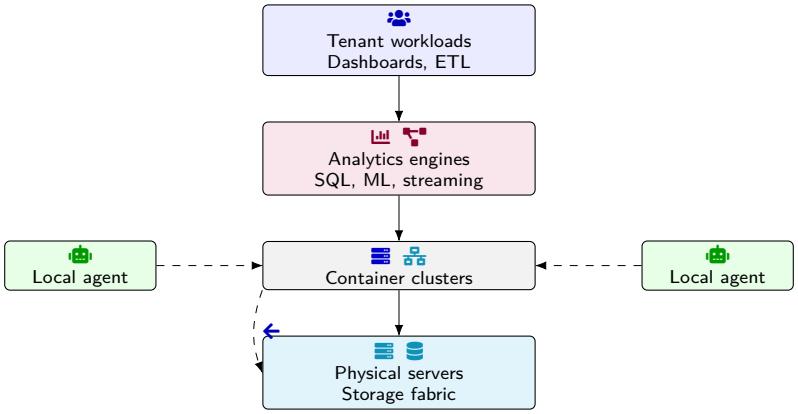

Figure 6: Enterprise analytics deployment, with tenant workloads mapped to shared analytics engines and container clusters, where local swarm agents attach to the cluster layer to drive elastic resource provisioning.

Table 2: Cluster configuration tiers for dynamic resource provisioning

Tier	Node type	vCPU per node	RAM per node (GB)
Ingest	General purpose	8	32
Compute A	Compute-optimized	16	64
Compute B	Compute-optimized	32	128
Storage	Memory-optimized	8	128
Control	Lightweight	4	16

Swarm coordination is introduced through an auxiliary set of swarm variables maintained by agents and propagated along a sparse communication graph [32]. Let this graph be described by a vertex set equal to the agent set and an edge set representing communication links between agents that share physical resources or are expected to experience correlated workloads. Each agent maintains a scalar or low-dimensional swarm state that summarizes its local congestion or resource pressure. A simple example is a congestion level computed from queue backlogs or latency deviations. At each step, agents exchange swarm states with neighbors and update their own swarm state using a consensus-like rule [33]. For a scalar swarm variable at agent, a typical update has the form

$$z_{t+1}^{i} = z_{t}^{i} + \gamma \sum_{j[34] \in \mathcal{N}_i} w_{ij} \left(z_{t}^{j} - z_{t}^{i} \right) \tag{11}$$

where the neighborhood set denotes the neighbors of agent in the communication graph, the weights are nonnegative and sum to one over neighbors for each agent, and the scalar is a consensus step size. This update drives the swarm variables toward a weighted average of local congestion signals, enabling distributed propagation of aggregate load information [35].

The swarm variables are integrated into the reinforcement learning process through the feature mapping and policy parameterization. For each agent, the feature vector includes its local swarm state and possibly smoothed differences between its swarm state and those of its neighbors. These features allow the action-value function to capture how scaling decisions should react not only to local utilization but also to the congestion level of nearby pools [36]. Policies are parameterized as stochastic mappings, for example using a softmax distribution over a discrete action set. Let the preference of agent for action given observation and swarm state be written as

$$h_{\psi^i}^{i}(o_t^i, z_t^i, [37]u^i) = \eta^i(o_t^i, z_t^i, [38]u^i)^{\top} \psi^i \tag{12}$$

where the vector is a policy feature mapping and the vector is a parameter vector. The policy assigns probabilities via

$$\pi_{\psi^i}^{i}(u^i | o_t^i, [39] z_t^i) = \frac{\exp(h_{\psi^i}^{i}(o_t^i, z_t^i, [40]u^i))}{\sum_{v \in \mathcal{A}_i} \exp(h_{\psi^i}^{i}(o_t^i, z_t^i, v))} \tag{13}$$

where the action set denotes the discrete scaling actions available to agent, such as adding or removing a small number of replicas [41].

The learning objective for each agent is to minimize its expected discounted or average cost, subject to the coupled dynamics induced by the joint policy of all agents. In the discounted formulation with factor in the interval between zero and one, the objective of agent is to minimize

$$J^i(\pi) = \mathbb{E}\Big[[42] \sum_{t=0}^{\infty} \beta^t c_t^i \Big] \tag{14}$$

where the expectation is taken with respect to the trajectory distribution induced by the joint policy and the dynamics. Since the environment is non-stationary from the perspective of any single agent due to changing policies of others, stability of learning is not guaranteed by default. Swarm coordination modifies this interaction by introducing structured coupling of policies through shared

Table 3: Swarm coordination mechanisms explored in the control layer

Mechanism	Communication pattern	Update frequency	Coordination signal
Gossip-based averaging	Random peer-to-peer	Every 5 steps	Local load and capacity hints
Leader election	Star topology	On major topology change	Global scaling intent
Neighborhood consensus	Ring lattice	Every 10 steps	Target utilization range
Broadcast heuristics	Central controller	Every step	Hard upper/lower bounds
Hierarchical clustering	Multi-level aggregation	Every 20 steps	Cluster-level resource budgets

Table 4: Autoscaling baselines used for comparison

Method	Description	Coordination	Adaptivity
Static provisioning	Fixed capacity per tier	None	None
Threshold-based scaling	Rule-based CPU thresholds	Local	Low
Reactive autoscaler	Built-in platform scaler	Local	Medium
Single-agent RL	Monolithic RL controller	Centralized	High
Heuristic swarm	Hand-crafted swarm rules	Decentralized	Medium
Proposed MARL swarm	Learned multi-agent swarm policy	Decentralized	High

features and consensus variables, which can reduce the effective non-stationarity experienced by each agent [43].

To update the value function parameters, temporal-difference methods with linear function approximation are used. For agent, after observing a transition from observation and local action to new observation and swarm state with observed cost, the temporal-difference error under a fixed target policy is computed as

$$\delta_t^i[44] = c_t^i + \beta \hat{V}^i(o_{t+1}^i, z_{t+1}^i) - \hat{V}^i(o_t^i, z_t^i) \tag{15}$$

where the value estimate is derived from the action-value function via

$$\hat{V}^i(o_t^i, z_t^i) = \sum_{u[45] \in \mathcal{A}_i} \pi_{\psi^i}^i(u|o_t^i, z_t^i) Q_{\theta^i}^i(o_t^i, u) \tag{16}$$

The parameter update for the action-value function then takes the form [46]

$$\theta_{t+1}^i = \theta_t^i - \alpha_t \delta_t^i g_t^i \tag{17}$$

where the step size satisfies standard stochastic approximation conditions and the eligibility vector is given by the gradient of the value estimate with respect to the parameters, typically equal to a suitable combination of features [47]. This linear update preserves the stability properties associated with temporal-difference learning with linear function approximation under appropriate conditions on the dynamics and policies.

Policy parameters are updated in an actor-critic fashion using approximate policy gradient estimates. For agent, the gradient of the objective with respect to the policy parameters can be approximated using the compatible features that relate the derivative of the log-policy to the action-value function estimates [48]. The policy parameter update is expressed as

$$\psi_{t+1}^i = \psi_t^i - \lambda_t \hat{g}_t^i \tag{18}$$

where the estimate is built from samples of [49]

$$\hat{g}_t^i \approx \nabla_{\psi^i} \log \pi_{\psi^i}^i(u_t^i|o_t^i, z_t^i)\, Q_{\theta^i}^i(o_t^i, [50]u_t^i) \tag{19}$$

and the step size is chosen analogously to. The presence of swarm variables in the policy features means that the gradient implicitly depends on the states and policies of neighboring agents through the consensus mechanism. Nevertheless, since swarm updates are linear and communication is local, the overall complexity of each update step scales with the degree of the communication graph rather than the total number of agents.

The communication graph and swarm update parameters influence how quickly congestion and resource pressure propagate across the platform [51]. A highly connected graph leads to rapid averaging of swarm variables but can blur local distinctions, while a sparse graph preserves locality but may slow down coordination. The consensus matrix associated with the weights can be used to analyze convergence of the swarm variables. If this matrix is doubly stochastic and the graph is connected, the consensus update converges to a common value equal to the average of initial swarm states [52]. The convergence rate is governed by the spectral gap, which in turn depends on the topology. By tuning the graph structure and weights, one can balance responsiveness and locality in the swarm-coordinated resource provisioning policy.

Linear Resource Provisioning and Optimization

The reinforcement learning framework operates on top of an underlying linear model of resource provisioning, which provides structure for both analysis and feature construction. The linear state evolution model combined with the stage cost defines a linear-quadratic-like control problem, except that the cost is piecewise linear and actions may be constrained to a finite set of increments [53]. To study the performance limits and to guide reward shaping, it is helpful to consider a relaxed optimization problem

Table 5: Service-level and cost metrics across autoscaling strategies

Method	SLA violations (%)	Normalized cost	Mean reaction time (s)
Static provisioning	14.2	1.35	–
Threshold-based	9.8	1.12	180
Reactive autoscaler	7.3	1.00	120
Single-agent RL	4.5	0.96	85
MARL swarm (proposed)	2.1	0.91	48

Table 6: Ablation study of swarm coordination components

Variant	Disabled component	Impact on SLA (% change)	Impact on utilization (% change)
Full model	None	0.0	0.0
No gossip	Gossip-based averaging	+36.4	-4.8
No leader election	Leader election	+19.7	+2.3
No neighborhood consensus	Neighborhood consensus	+24.1	-3.5
No global coordinator	Global coordinator	+41.8	+6.2
Independent agents	All coordination	+58.9	-7.6

in which actions are allowed to vary continuously within the feasible set and disturbances are approximated by their average effect.

In the relaxed setting, the goal is to find a stationary allocation vector that minimizes the long-run average cost subject to capacity and service level constraints. Under the assumption that the linear dynamics stabilize around a fixed point for constant actions, steady-state conditions can be written as [54]

$$x^\star = Ax^\star + Bu^\star + \bar{w} \tag{20}$$

where the constant vectors denote the steady-state state, action, and average disturbance. Solving for the steady-state state gives

$$(I - A)x^\star = Bu^\star + \bar{w} \tag{21}$$

Assuming the matrix is invertible and stable in the sense that its spectral radius is less than one, the steady-state state is expressed as

$$x^\star = (I - A)^{-1}(Bu^\star + \bar{w}) \tag{22}$$

Service level constraints in steady state then become linear inequalities in the action vector once this expression is substituted into the performance equations [55].

The relaxed steady-state optimization problem can be written as a linear program by introducing auxiliary variables for constraint violations and using linear approximations for performance metrics. For instance, one may define variables representing upper bounds on latency proxies, and then impose constraints tying these variables to linear combinations of the state. The objective is to minimize a weighted sum of action magnitudes and constraint violation variables [56]. A simplified version of the linear program takes the form

$$\min_{u,v} \quad c^\top u + r^\top v \tag{23}$$

$$\text{s.t.} \quad Fu + Gv \le h \tag{24}$$

where the vector collects action costs, the vector contains penalty coefficients, and the matrices and vector encode capacity, steady-state, and service level constraints. The variables represent slack or violation margins that allow the linear program to remain feasible even when strict constraint satisfaction is impossible under the given average workload.

The optimal solution of the relaxed linear program provides a baseline allocation that the reinforcement learning agents can use as a reference [57]. In particular, the solution indicates how resources should be distributed across pools under average conditions to trade off cost and performance penalties. Deviations from this allocation can then be interpreted as responses to transient workload fluctuations and stochastic variability. This interpretation suggests a decomposition in which the action of each agent is written as the sum of a baseline component derived from the linear program and a corrective component generated by the learned policy, which attempts to compensate for short-term variations and prediction errors [58].

To integrate this structure into the multi-agent reinforcement learning framework, features are constructed that encode both the deviation of the current allocation from the baseline and the estimated marginal cost of resources implied by the dual variables of the linear program. The dual variables associated with capacity constraints can be viewed as shadow prices for resource units in each pool, indicating how the objective would change if capacity were slightly increased. If these prices are approximated or periodically recomputed, they can be included in the feature vectors to bias the learned policies toward actions that align with the underlying linear optimization problem.

Consider an agent controlling a particular pool [59]. Let the scalar denote the shadow price of one unit of CPU capacity in that pool, and similarly for memory and I/O. The feature vector for this agent can be extended to include terms such as the product of utilization and the

Table 7: Key hyperparameters for training the multi-agent reinforcement learning swarm

Parameter	Symbol	Value	Description
Discount factor	γ	0.99	Long-term reward weighting
Learning rate	α	3×10^{-4}	Actor-critic optimizer step size
Target update rate	τ	0.005	Soft target network updates
Batch size	B	512	Experience replay batch size
Replay buffer size	$\lvert\mathcal{D}\rvert$	10^6	Centralized experience buffer
Coordination loss weight	λ_c	0.3	Strength of swarm regularization
Exploration temperature	T	0.7	Entropy-based exploration

corresponding shadow price, capturing the idea that scaling decisions should depend both on how heavily resources are used and on their marginal cost. In linear terms, if the agent considers a small scaling action in CPU allocation, the immediate change in the relaxed objective can be approximated as [60]

$$\Delta c^i \approx \mu^i_{cpu} \Delta u^i_{cpu} \tag{25}$$

where the scalar represents the shadow price and the scalar is the proposed change in CPU capacity. Similar expressions apply to other resource dimensions [61]. These approximations do not require exact dual solutions at every decision step; rather, they provide directional guidance that can regularize the learned policies.

Stability properties of the linear model are important for ensuring that reinforcement learning does not drive the system toward unstable configurations. The matrix describing the endogenous dynamics should have a spectral radius strictly less than one for the linear approximation to be meaningful over extended horizons [62]. In addition, the effect of actions encoded in the matrix should respect physical constraints: increasing capacity should not increase queue lengths in the linear approximation. These conditions can be encoded as sign and magnitude constraints on the entries of the matrices. For example, one may require that certain entries be nonpositive to ensure that increased capacity reduces congestion metrics in the state.

When actions are restricted to a discrete set of scaling increments, the linear dynamics still provide insight through the concept of local controllability [63]. A pool is locally controllable in a given region of the state space if small adjustments in capacity can move the state in directions that reduce cost and violation penalties. Under the linear model, this property can be analyzed by examining the rank and structure of the matrix restricted to the components of the state corresponding to queue lengths and utilization. If the controllable subspace is sufficiently rich, reinforcement learning agents can, in principle, discover policies that steer the system toward regions of the state space with lower cost [64].

Finally, the interaction between swarm coordination and the linear provisioning model can be expressed in terms of aggregate variables. For instance, let the swarm state at each agent estimate a linear functional of the local state, such as a weighted sum of utilization and queue lengths. The consensus update then approximates a distributed averaging of this linear functional across the graph. Under suitable conditions on the weights, the swarm variables converge to a value proportional to the average congestion across the platform or across a region of the cluster [65]. This global congestion estimate influences local actions and effectively implements a linear feedback term that couples the dynamics of different pools, mitigating the risk of localized overreaction and oscillation.

Experimental Evaluation

The behavior of the swarm-coordinated multi-agent reinforcement learning framework is evaluated in a simulated enterprise analytics platform that captures key structural features of real deployments. The simulation environment models a cluster with multiple resource pools, each comprising a fixed number of physical nodes and hosting several analytic services [66]. Workloads consist of multiple traffic classes with distinct latency requirements and resource profiles. Arrival processes combine periodic patterns to capture diurnal effects and superimposed bursts representing business events and batch pipeline triggers. Service times depend on resource allocations, with diminishing returns as capacity approaches saturation.

The linear state evolution model is calibrated using queueing-theoretic approximations derived from the simulated service configurations [67]. For each pool, the simulation tracks queue lengths, utilization levels, and response time proxies as a function of the allocated capacity and arrival rates. Linearization is performed around operating points corresponding to moderate utilization, and the resulting matrices are used to define the dynamics for the reinforcement learning agents. To account for nonlinear effects at high utilization, disturbances are injected that capture the discrepancy between the linear approximation and the actual simulated behavior [68]. This creates a scenario in which the learning agents must compensate for model mismatch while exploiting the structural information encoded in the linear model.

Agents are positioned at the level of resource pools. Each agent observes local metrics including smoothed CPU and memory utilization, queue length indicators for each traffic class served in its pool, and the difference

between observed latencies and their targets. In addition, each agent maintains a swarm state that is updated via consensus with neighbors according to the communication graph [69]. The underlying graph is chosen to reflect physical proximity and shared bottlenecks, with agents connected when their pools share underlying network links or storage systems. The step size and weights in the consensus update are tuned to achieve stable convergence while remaining responsive to changes in congestion.

The agents use linear action-value functions and softmax policies as described in the previous sections [70]. Features include normalized utilization, queue ratios, deviation of latency proxies from targets, deviations of allocations from the baseline determined by the relaxed linear program, and swarm variables. The action space for each agent consists of discrete scaling steps for CPU and memory allocation: increasing or decreasing capacity by fixed increments within the feasible set. The reward signal is constructed as the negative of the local stage cost, combining resource usage and penalties for violating service level constraints. Discounted and average-cost formulations are both considered, with discount factors close to one to emphasize long-run behavior [71].

To assess the impact of swarm coordination, several configurations are compared. A first configuration uses independent reinforcement learning agents without swarm variables or communication. Each agent bases its policy solely on local observations [72]. A second configuration activates swarm coordination by including swarm variables in the feature set and enabling consensus updates along the communication graph. A third configuration employs a centralized reinforcement learning controller that observes aggregate state and selects joint actions, subject to the same action space as the decentralized agents. This central controller is trained using a similar actor-critic scheme but operates in a higher-dimensional state-action space.

The experimental scenarios cover a range of workload patterns and platform configurations [73]. In one set of experiments, the cluster experiences slow diurnal variations with moderate bursts, and service level constraints are relatively loose. In this regime, all control strategies are expected to maintain acceptable performance, but resource utilization levels differ. In another set of experiments, abrupt spikes are introduced in specific traffic classes in certain pools, while others remain lightly loaded [74]. This setting tests how effectively the different controllers can redirect capacity toward hot spots without violating constraints in other parts of the cluster. A final set of experiments introduces correlated bursts across many pools, creating global stress on the platform and forcing trade-offs between classes.

Metrics examined include average and tail latency proxies per traffic class, the fraction of time that service level constraints are violated, total resource consumption over time, and measures of policy stability such as the frequency and amplitude of capacity adjustments. In addition, the convergence behavior of the learning algorithms is monitored through the evolution of value function parameters, policy entropy, and swarm variables [75]. The interplay between swarm coordination and learning is observed by tracking how quickly local policies adapt to changes in global congestion patterns and how often oscillatory behaviors arise.

Across the scenarios examined, swarm-coordinated multi-agent reinforcement learning exhibits a consistent pattern. Compared to independent agents, the swarm-coordinated agents tend to produce more synchronized yet smooth capacity adjustments across neighboring pools [76]. When a burst occurs in one pool, the corresponding agent increases its swarm state, which then propagates through the communication graph. Neighboring agents interpret this increase as evidence of elevated congestion in the region and are more conservative in scaling down their own capacity, even if their local metrics do not yet exceed thresholds. This behavior reduces the risk that capacity is prematurely reclaimed from lightly loaded pools that are about to experience spillover traffic.

Relative to the centralized controller, the swarm-coordinated agents show similar adherence to service level constraints but differ in how they realize capacity adjustments [77]. The centralized controller can, in principle, compute joint actions that optimize global criteria, but it operates in a larger state-action space and thus requires more samples and computation per update. In the experiments, centralized learning converges more slowly and can become sensitive to model mismatch in the linear approximation, particularly when nonlinearities in utilization-latency relationships become pronounced. The decentralized swarm-coordinated agents, by contrast, leverage local structure and operate on lower-dimensional observations, which improves sample efficiency while relying on swarm variables to share essential congestion information [78].

The evaluation also highlights the influence of the communication graph topology on performance. For highly connected graphs, swarm variables converge quickly across the entire cluster, causing agents to respond in a relatively uniform manner to global congestion. This can be beneficial when workloads are strongly correlated across pools, as it promotes coordinated reductions in capacity during low-demand periods and coordinated preservation of headroom during high demand. However, in scenarios with localized hot spots, overly connected graphs can spread congestion signals too widely, leading to conservative behavior even in pools that remain lightly loaded [79]. Sparser graphs concentrate swarm coordination within regions of the platform, enabling more differentiated responses at the cost of slower propagation of congestion signals.

Sensitivity analysis with respect to consensus step size and swarm feature scaling indicates that there is a range of parameters for which swarm coordination stabilizes learning

and improves resource utilization without introducing oscillations. If consensus updates are too aggressive or swarm variables are weighted too heavily in the feature vectors, the agents may overreact to transient fluctuations in swarm states, amplifying noise and creating correlated oscillations in capacity allocations [80]. When consensus updates are too weak, swarm variables change slowly and provide limited benefit over purely local observations. Intermediate settings achieve a balance, allowing swarm signals to influence action selection while preserving the role of local metrics.

Conclusion

This paper has examined dynamic resource provisioning in enterprise analytics platforms through a combined perspective of multi-agent reinforcement learning, swarm coordination, and linear resource modeling. The platform is represented as a set of interacting resource pools, each controlled by an agent that observes local metrics, exchanges swarm variables with neighbors, and selects scaling actions within a constrained action set [81]. A linear approximation of the state dynamics and performance metrics provides structure for value function approximation and steady-state optimization, while swarm coordination offers a decentralized way to propagate congestion information across the cluster.

The formulation of the problem as a Markov game with linear dynamics and piecewise linear cost enables the use of linear function approximation for action-value functions and actor-critic policy updates. The inclusion of swarm variables in the feature representations couples agents through a consensus mechanism on a communication graph, influencing learning and action selection without requiring centralized control [82]. The relaxed linear program describing steady-state allocations informs baseline resource distributions and suggests feature constructions based on approximate shadow prices, which guide agents toward policies that align with the underlying optimization problem.

Experimental evaluation in a simulated analytics platform indicates that swarm-coordinated multi-agent reinforcement learning can achieve resource utilization and service level adherence that are competitive with a centralized reinforcement learning controller, while maintaining a decentralized architecture. Compared to independent learning agents, the swarm-coordinated agents exhibit more stable policies and reduced oscillations in capacity allocations, particularly under bursty and correlated workloads. The communication graph topology and swarm update parameters play a significant role in shaping this behavior, creating trade-offs between responsiveness to global congestion and sensitivity to local conditions [83].

Several limitations of the present study suggest directions for further work. The linear approximation of resource and performance dynamics simplifies the underlying queueing behavior and may lose accuracy in regimes with high utilization or complex workloads. Extending the framework to incorporate richer models, possibly through piecewise linear or nonlinear function approximations, could improve fidelity while preserving analytical tractability [84]. The current formulation also assumes fixed communication graphs and static swarm update rules; adapting these structures over time based on observed workload correlations and performance outcomes may yield additional gains. Finally, exploring integration with existing orchestration systems and validating the framework on production traces would provide further insight into its practical behavior and robustness. the combination of multi-agent reinforcement learning with swarm coordination and linear resource modeling provides a structured approach to dynamic provisioning in enterprise analytics platforms. It leverages local decision making, lightweight communication, and linear analytical tools to balance resource efficiency and service level compliance under stochastic and time-varying workloads, offering a basis for further theoretical and empirical investigation [85].

Conflict of interest

Authors state no conflict of interest.

References

[1] T. Soule and R. B. Heckendorn, "Gecco (companion) - developmental scalable hierarchies for multi-agent swarms," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, ACM, Jul. 12, 2011, pp. 207–208. DOI: 10.1145/2001858.2001974

[2] M. Winikoff, N. Desai, and A. Liu, "Principles and practice of multi-agent systems - principles and practice of multi-agent systems," *Multiagent and Grid Systems*, vol. 8, no. 2, pp. 125–126, May 4, 2012. DOI: 10.3233/mgs-2012-0188

[3] J. Shao, J. Wang, and L. Yang, "Cscw - a production monitoring and data processing system for the textile enterprise based on multi-agent," in *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, ACM, Mar. 19, 2011, pp. 713–716. DOI: 10.1145/1958824.1958952

[4] R. Chandrasekar and S. Misra, "Using zonal agent distribution effectively for routing in mobile ad hoc networks," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 3, no. 2, pp. 82–89, 2008.

[5] P. Lin, Z. Li, Y. Jia, and M. Sun, "High-order multi-agent consensus with dynamically changing topologies and time-delays," *IET Control Theory & Applications*, vol. 5, no. 8, pp. 976–981, May 19, 2011. DOI: 10.1049/iet-cta.2009.0649

[6] J. Giampapa, O. H. Juarez-Espinosa, and K. Sycara, "Agents - configuration management for multi-agent systems," in *Proceedings of the fifth international conference on Autonomous agents*, ACM, May 28, 2001, pp. 230–231. DOI: 10.1145/375735.376295

[7] A. Murciano, J. del R. Millán, and J. Zamora, "Specialization in multi-agent systems through learning," *Biological cybernetics*, vol. 76, no. 5, pp. 375–382, May 30, 1997. DOI: 10.1007/s004220050351

[8] S. H. Kukkuhalli, "Optimizing snowflake enterprise data platform cost through predictive analytics and query performance optimization," *IJSAT-International Journal on Science and Technology*, vol. 15, no. 4, 2024.

[9] T. Araújo and F. Louçã, "Modeling a multi-agents system as a network: A metaphoric exploration of the unexpected," *International Journal of Agent Technologies and Systems*, vol. 1, no. 4, pp. 17–29, Oct. 1, 2009. DOI: 10.4018/jats.2009100102

[10] S. K. Amirgholipour, "Multi agent electronic cargo terminal system," *Indian Journal of Science and Technology*, vol. 7, no. 4, pp. 430–438, Apr. 20, 2014. DOI: 10.17485/ijst/2014/v7i4.5

[11] P.-M. Ricordel and Y. Demazeau, "Esaw - from analysis to deployment: A multi-agent platform survey," *Lecture Notes in Computer Science*, pp. 93–105, Dec. 15, 2000. DOI: 10.1007/3-540-44539-0_7

[12] M. Duvigneau, D. Moldt, and H. Rölke, *AOSE - Concurrent architecture for a multi-agent platform*. Germany: Springer Berlin Heidelberg, Mar. 14, 2003. DOI: 10.1007/3-540-36540-0_5

[13] K. Busaki, Y. Iijima, and S. Konno, "Multi-agent system learning support software with fighting games," *International Journal of Energy, Information and Communications*, vol. 5, no. 5, pp. 13–22, Oct. 31, 2014. DOI: 10.14257/ijeic.2014.5.5.02

[14] G. Samigulina, A. Nyussupov, and A. Shayakhmetova, "Multi-agent smart-system of distance learning for people with vision disabilities," in Germany: Springer International Publishing, May 28, 2017, pp. 154–166. DOI: 10.1007/978-3-319-59451-4_16

[15] L. Wang and Z. Liu, "Robust consensus of multi-agent systems with bounded disturbances," in InTech, Apr. 1, 2011. DOI: 10.5772/15470

[16] L. Drumond, R. Girardi, and A. Leite, "Icail - architectural design of a multi-agent recommender system for the legal domain," in *Proceedings of the 11th international conference on Artificial intelligence and law*, ACM, Jun. 4, 2007, pp. 183–187. DOI: 10.1145/1276318.1276352

[17] C. Ramachandran, R. Malik, X. Jin, J. Gao, K. Nahrstedt, and J. Han, "Videomule: A consensus learning approach to multi-label classification from noisy user-generated videos," in *Proceedings of the 17th ACM international conference on Multimedia*, 2009, pp. 721–724.

[18] H. Osano, "Implementation of multi-agent incentive contracts with the principal's renegotiation offer," *Review of Economic Design*, vol. 4, no. 2, pp. 161–177, Jun. 1, 1999. DOI: 10.1007/s100580050031

[19] T. R. Burns and E. Roszkowska, "Cognition and multi-agent interaction: Social judgment in multi-agent systems," in Cambridge University Press, Dec. 5, 2005, pp. 409–416. DOI: 10.1017/cbo9780511610721.018

[20] D. Liu, A. Hu, and D. Zhao, "Distributed consensus of multi-agent networks via event-triggered pinning control," *Asian Journal of Control*, vol. 19, no. 2, pp. 614–624, Sep. 15, 2016. DOI: 10.1002/asjc.1389

[21] G. Wen, H.-T. Zhang, W. Yu, Z. Zuo, and Y. Zhao, "Coordination tracking of multi-agent dynamical systems with general linear node dynamics," *International Journal of Robust and Nonlinear Control*, vol. 27, no. 9, pp. 1526–1546, Jan. 23, 2017. DOI: 10.1002/rnc.3753

[22] S. Turgay and F. Yaman, "Intelligent query answering mechanism in multi agent systems," in IGI Global, Jan. 18, 2011. DOI: 10.4018/9781599048499.ch136

[23] Z. Xue, Q. Dong, X. Fan, Q. Jin, H. Jian, and J. Liu, "Fuzzy logic-based model that incorporates personality traits for heterogeneous pedestrians," *Symmetry*, vol. 9, no. 10, pp. 239–, Oct. 20, 2017. DOI: 10.3390/sym9100239

[24] J. Xuesong, T. Sun, T. Qiaoyun, and J. Wang, "Flexible workshop scheduling optimization based on multi-agent technology," *International Journal of Hybrid Information Technology*, vol. 9, no. 5, pp. 303–310, May 31, 2016. DOI: 10.14257/ijhit.2016.9.5.25

[25] S. Martin, *Multi-agent flocking under topological interactions*, Jan. 1, 2013. DOI: 10.48550/arxiv.1311.6046

[26] A. Chaib, I. Boussebough, and A. Chaoui, "Adaptive service composition in an ambient environment with a multi-agent system," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 2, pp. 367–380, Jun. 1, 2017. DOI: 10.1007/s12652-017-0510-8

[27] L. Teng, N. Liu, R. Liu, and L. Lu, "A study on service-based multi-agent geospatial data sharing method," *SPIE Proceedings*, vol. 6754, pp. 265–272, Jun. 10, 2007. DOI: 10.1117/12.764621

[28] F.-A. Parand, N. M. Charkari, and S. Afrasyabia, "Response fusion in multi-agent environment," *American Journal of Applied Sciences*, vol. 6, no. 1, pp. 57–63, Jan. 1, 2009. DOI: 10.3844/ajas.2009.57.63

[29] A. U. Frank, S. Bittner, and M. Raubal, "Cosit - spatial and cognitive simulation with multi-agent systems," in Germany: Springer Berlin Heidelberg, Nov. 13, 2001, pp. 124–139. DOI: 10.1007/3-540-45424-1_9

[30] V. Vijaykumar, R. Chandrasekar, and T. Srinivasan, "An obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs," in *2006 IEEE Conference on Cybernetics and Intelligent Systems*, 2006, pp. 1–6. DOI: 10.1109/ICCIS.2006.252326

[31] R. Frozza and L. O. Alvares, "Coordination - criteria for the analysis of coordination in multi-agent applications," in Germany: Springer Berlin Heidelberg, Mar. 14, 2002, pp. 158–165. DOI: 10.1007/3-540-46000-4_17

[32] N. Mesbahi, O. Kazar, S. Benharzallah, and M. Zoubeidi, "A cooperative multi-agent approach-based clustering in enterprise resource planning," *International Journal of Knowledge and Systems Science*, vol. 6, no. 1, pp. 34–45, Jan. 1, 2015. DOI: 10.4018/ijkss.2015010103

[33] W. Chen, Z. Tengfei, and L. Xuxin, "A multi-agent based coordinated control strategy for distributed loads in microgrid," *International Journal of Control and Automation*, vol. 9, no. 8, pp. 219–232, Aug. 31, 2016. DOI: 10.14257/ijca.2016.9.8.21

[34] V. Yazdanpanah and M. Dastani, "Prima - distant group responsibility in multi-agent systems," in Germany: Springer International Publishing, Aug. 10, 2016, vol. 9862, pp. 261–278. DOI: 10.1007/978-3-319-44832-9_16

[35] J. Magott, "Estimation of mean response time of multi-agent systems using petri nets," in I-Tech Education and Publishing, Feb. 1, 2008. DOI: 10.5772/5324

[36] Q. Liu, Y. Zhao, and L. Cheng, "Isnn - continuous-time multi-agent network for distributed least absolute deviation," in Germany: Springer International Publishing, Nov. 19, 2015, pp. 436–443. DOI: 10.1007/978-3-319-25393-0_48

[37] A. A. Abood, A. N. Abdalla, and S. K. Avakian, "The application of multi-agent technology on transient stability assessment of iraqi super grid network," *American Journal of Applied Sciences*, vol. 5, no. 11, pp. 1494–1498, Nov. 1, 2008. DOI: 10.3844/ajassp.2008.1494.1498

[38] V. Kumar, B. K. Kaushik, and H. Banka, "Improved multi-agent reinforcement learning for minimizing traffic waiting time," *International Journal of Computer Applications*, vol. 81, no. 9, pp. 30–34, Nov. 15, 2013. DOI: 10.5120/14043-2205

[39] T. Ren, Y.-f. Wang, M.-m. Liu, C.-j. Li, and Y. Liu, "The consensus of nonlinear multi-agent system with switching topologies and communication failure," *Journal of Intelligent & Fuzzy Systems*, vol. 30, no. 2, pp. 1199–1206, Nov. 2, 2015. DOI: 10.3233/ifs-151844

[40] R. Kouwenberg and R. C. J. Zwinkels, "Endogenous price bubbles in a multi-agent system of the housing market," *PloS one*, vol. 10, no. 6, pp. 1–10, Jun. 24, 2015. DOI: 10.1371/journal.pone.0129070

[41] F. Rateb, B. Pavard, N. Bellamine-BenSaoud, J. J. Merelo, and M. I. G. Arenas, "Modeling malaria with multi-agent systems," *International Journal of Intelligent Information Technologies*, vol. 1, no. 2, pp. 17–27, Apr. 1, 2005. DOI: 10.4018/jiit.2005040102

[42] M. Maleković and M. Čubrilo, "Incorporating infatuation in multi-agent systems," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 10, no. 4, pp. 517–521, Jul. 20, 2006. DOI: 10.20965/jaciii.2006.p0517

[43] T. Srinivasan, R. Chandrasekar, V. Vijaykumar, V. Mahadevan, A. Meyyappan, and M. Nivedita, "Exploring the synergism of a multiple auction-based task allocation scheme for power-aware intrusion detection in wireless ad-hoc networks," in *2006 10th IEEE Singapore International Conference on Communication Systems*, IEEE, 2006, pp. 1–5.

[44] J. Liang-Hao, L. Xiao-feng, and C. Xin, "Pinning consensus analysis of multi-agent networks with arbitrary topology," *Chinese Physics B*, vol. 22, no. 9, pp. 090 202–, Sep. 25, 2013. DOI: 10.1088/1674-1056/22/9/090202

[45] X. Li and Z. Dong, "Platform-level distributed warfare model-based on multi-agent system framework," *Defence Science Journal*, vol. 62, no. 3, pp. 180–186, May 3, 2012. DOI: 10.14429/dsj.62.964

[46] H. Du and R. Jia, "Synchronization of a class of nonlinear multi-agent systems with sampled-data information," *Nonlinear Dynamics*, vol. 82, no. 3, pp. 1483–1492, Jul. 15, 2015. DOI: 10.1007/s11071-015-2255-2

[47] S. Jyothi and A. Damodar, "Cryptographic protocol based laurel system for multi agent distributed communication," *International Journal of Computer Engineering in Research Trends*, vol. 3, no. 9, pp. 500–, Sep. 1, 2016. DOI: 10.22362/ijcert/2016/v3/i9/48899

[48] C. Yang et al., "Consensus for non-linear multi-agent systems modelled by pdes based on spatial boundary communication," *IET Control Theory & Applications*, vol. 11, no. 17, pp. 3196–3200, Sep. 27, 2017. DOI: 10.1049/iet-cta.2017.0479

[49] B. Qi, X. Lou, and B. Cui, "Containment control of second-order multi-agent systems with directed topology and time-delays," *Kybernetes*, vol. 43, no. 8, pp. 1248–1261, Aug. 26, 2014. DOI: 10.1108/k-06-2013-0113

[50] F. Enembreck and J.-P. A. Barthès, "Mais – un système multi-agents pour la recherche d'information sur le web," *Document numérique*, vol. 8, no. 3, pp. 83–106, Sep. 1, 2004. DOI: 10.3166/dn.8.3.83-106

[51] F. Bergenti and A. Poggi, "Esaw - exploiting uml in the design of multi-agent systems," *Lecture Notes in Computer Science*, pp. 106–113, Dec. 15, 2000. DOI: 10.1007/3-540-44539-0_8

[52] Q. Wang and Y. Wang, "Cluster synchronization of a class of multi-agent systems with a bipartite graph topology," *Science China Information Sciences*, vol. 57, no. 1, pp. 1–11, Nov. 30, 2012. DOI: 10.1007/s11432-012-4689-1

[53] K. Ono, T. Hata, T. Maetani, M. Harao, and K. Hirata, "Jsai workshops - development of a multi-agent based generic traffic simulator," *Lecture Notes in Computer Science*, pp. 249–260, Jun. 29, 2006. DOI: 10.1007/11780496_28

[54] J. Zhang, *Multi-agent-based production control*, Apr. 26, 2017. DOI: 10.1002/9781118890073.ch7

[55] B. Sasikumar and V. Vasudevan, "Multi agent system based tcp for wireless networks," *International Journal of Computer Applications*, vol. 27, no. 6, pp. 45–50, Aug. 31, 2011. DOI: 10.5120/3301-4509

[56] T. Srinivasan, V. Vijaykumar, and R. Chandrasekar, "An auction based task allocation scheme for power-aware intrusion detection in wireless ad-hoc networks," in *2006 IFIP International Conference on Wireless and Optical Communications Networks*, IEEE, 2006, 5–pp.

[57] T. Logenthiran and D. Srinivasan, "Management of distributed energy resources using intelligent multi-agent system," in IGI Global, Jan. 18, 2011. DOI: 10.4018/9781605668987.ch012

[58] Z. Wang, L. Xia, and Y. Wang, "Application of multi-agent and genetic algorithm in network reconfiguration of ship power system," *Elektronika ir Elektrotechnika*, vol. 18, no. 9, pp. 7–10, Nov. 9, 2012. DOI: 10.5755/j01.eee.18.9.2795

[59] E. Majd and V. Balakrishnan, "A reputation-oriented trust model for multi-agent environments," *Industrial Management & Data Systems*, vol. 116, no. 7, pp. 1380–1396, Aug. 8, 2016. DOI: 10.1108/imds-06-2015-0256

[60] V. Marik, J. Mueller, and M. Pechoucek, *Multi-Agent Systems and Applications III - Multi-Agent Systems and Applications III*. Germany: Springer Berlin Heidelberg, May 27, 2003. DOI: 10.1007/3-540-45023-8

[61] M. Dastani, J. Dix, and P. Novák, "The second contest on multi-agent systems based on computational logic," in Springer Berlin Heidelberg, May 14, 2007, pp. 266–283. DOI: 10.1007/978-3-540-69619-3_15

[62] G. Yang, V. Kapila, and R. Vaidyanathan, "A dynamic-programming-styled algorithm for a class of multi-agent optimal task assignment," in *Dynamic Systems and Control*, American Society of Mechanical Engineers, Nov. 11, 2001, pp. 293–299. DOI: 10.1115/imece2001/dsc-24536

[63] D. Weyns, "Middleware for distributed multi-agent systems," in Springer Berlin Heidelberg, Feb. 5, 2010, pp. 93–122. DOI: 10.1007/978-3-642-01064-4_5

[64] X. Dong, Q. Li, Q. Zhao, and Z. Ren, "Time-varying group formation analysis and design for general linear multi-agent systems with directed topologies," *International Journal of Robust and Nonlinear Control*, vol. 27, no. 9, pp. 1640–1652, Sep. 8, 2016. DOI: 10.1002/rnc.3650

[65] K. S. Ali and J. L. Shumaker, "Hardware in the loop simulator for multi agent unmanned aerial vehicles environment," *American Journal of Engineering and Applied Sciences*, vol. 6, no. 2, pp. 172–177, Feb. 1, 2013. DOI: 10.3844/ajeassp.2013.172.177

[66] Z. Li, X. Liu, M. Fu, and L. Xie, "Global H_∞ consensus of multi-agent systems with lipschitz nonlinear dynamics," *IET Control Theory & Applications*, vol. 6, no. 13, pp. 2041–2048, Sep. 6, 2012. DOI: 10.1049/iet-cta.2011.0555

[67] J. Y. Halpern and R. Pucella, "Acm conference on computer and communications security - on the relationship between strand spaces and multi-agent systems," in *Proceedings of the 8th ACM conference on Computer and Communications Security*, ACM, Nov. 5, 2001, pp. 106–115. DOI: 10.1145/501983.501999

[68] J. J. Arockiaraj, "Fuzzy max-min composition in quality specifications of multi-agent system," *INTERNATIONAL JOURNAL OF COMPUTING ALGORITHM*, vol. 3, no. 2, pp. 101–106, Dec. 1, 2014. DOI: 10.20894/ijcoa.101.003.002.001

[69] V. Vijaykumar, R. Chandrasekar, and T. Srinivasan, "An ant odor analysis approach to the ant colony optimization algorithm for data-aggregation in wireless sensor networks," in *2006 International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, 2006, pp. 1–4.

[70] V. Werneck, R. M. E. M. da Costa, and L. M. Cysneiros, "Modelling multi-agent system using different methodologies," in InTech, Apr. 1, 2011. DOI: 10.5772/14792

[71] M. Arora and M. S. Devi, "Design of multi agent system for resource allocation and monitoring," in IGI Global, Apr. 12, 2012, pp. 1–10. DOI: 10.4018/978-1-4666-1565-6.ch001

[72] K. Ayyoub, F. Mokhati, and M. Badri, "Towards a new approach for controlling the reorganization process of multi-agent systems," in IGI Global, Nov. 9, 2017, pp. 2068–2087. DOI: 10.4018/978-1-5225-3923-0.ch088

[73] E. A. Olajubu, G. A. Aderounmu, and E. R. Adagunodo, "Network resources management in a multi-agent system: A simulative approach," *South African Journal of Science*, vol. 106, no. 9/10, pp. 1–6, Sep. 17, 2010. DOI: 10.4102/sajs.v106i9/10.322

[74] K. M. Carley, E. Malloy, and N. Altman, "Multi-agent modeling of biological and chemical threats," in Springer US, Oct. 19, 2010, pp. 361–380. DOI: 10.1007/978-1-4419-6892-0_16

[75] B. Wang and Y. Sun, "Consensus analysis of heterogeneous multi-agent systems with time-varying delay," *Entropy*, vol. 17, no. 6, pp. 3631–3644, Jun. 2, 2015. DOI: 10.3390/e17063631

[76] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with jade," in Germany: Springer Berlin Heidelberg, Aug. 24, 2001, pp. 89–103. DOI: 10.1007/3-540-44631-1_7

[77] W. Liu, C. Yang, F. Deng, and J. Liang, "Synchronization of general linear multi-agent systems with measurement noises," *Asian Journal of Control*, vol. 19, no. 2, pp. 510–520, Aug. 3, 2016. DOI: 10.1002/asjc.1357

[78] S. Akdemir and N. Erdogan, "Epia - multi-agent based file replication and consistency management," in Germany: Springer International Publishing, Aug. 9, 2017, pp. 727–738. DOI: 10.1007/978-3-319-65340-2_59

[79] J. Wu and R. Tzoneva, "A multi-agent system architecture for coordination of the real-time control functions in complex industrial systems," *International Journal of Computers Communications & Control*, vol. 6, no. 4, pp. 761–778, Dec. 1, 2011. DOI: 10.15837/ijccc.2011.4.2108

[80] H. R. Jordan, J. Treanor, D. Lillis, M. Dragone, R. W. Collier, and G. M. P. O'Hare, "Af-able in the multi agent contest 2009," *Annals of Mathematics and Artificial Intelligence*, vol. 59, no. 3, pp. 389–409, May 7, 2010. DOI: 10.1007/s10472-010-9180-3

[81] D. Kuiper and R. Z. Wenkstern, "Agent vision in multi-agent based simulation systems," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 2, pp. 161–191, Feb. 22, 2014. DOI: 10.1007/s10458-014-9250-8

[82] C. Ramachandran, S. Misra, and M. Obaidat, "On evaluating some agent-based intrusion detection schemes in mobile ad-hoc networks," in *Proceedings of the SPECTS 2007*, San Diego, CA, Jul. 2007, pp. 594–601.

[83] M. Lin, L. Ou, M. Wang, L. Zhang, and L. Yu, "The local control scheme for switching consensus value in multi-agent systems," *Journal of Control and Decision*, vol. 2, no. 3, pp. 185–202, Jun. 30, 2015. DOI: 10.1080/23307706.2015.1057546

[84] L. Brim, D. Gilbert, J.-M. Jacquet, and M. Kretínský, "Sofsem - multi-agent systems as concurrent constraint processes," in Germany: Springer Berlin Heidelberg, Oct. 24, 2001, pp. 201–210. DOI: 10.1007/3-540-45627-9_17

[85] F. Blatt, M. Becker, and H. Szczerbicka, "Mates - analysing the cost-efficiency of the multi-agent flood algorithm in search and rescue scenarios," in Germany: Springer International Publishing, Sep. 8, 2016, pp. 147–154. DOI: 10.1007/978-3-319-45889-2_11